PCT/JP03/16916
26.12.03

日本国特許庁
JAPAN PATENT OFFICE

JP03/16916

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　１月　６日

出願番号
Application Number: 特願２００３－０００２７５

[ST. 10/C]: [ＪＰ２００３－０００２７５]

出願人
Applicant(s): 松下電器産業株式会社

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

２００４年　２月　５日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

出証番号　出証特２００４－３００６４９８

【書類名】　特許願
【整理番号】　2968140085
【提出日】　平成15年 1月 6日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 9/45
【発明者】
【住所又は居所】　愛知県名古屋市中区栄２丁目６番１号白川ビル別館５階　株式会社松下電器情報システム名古屋研究所内
【氏名】　河合　正樹
【発明者】
【住所又は居所】　愛知県名古屋市中区栄２丁目６番１号白川ビル別館５階　株式会社松下電器情報システム名古屋研究所内
【氏名】　川本　琢二
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
【氏名】　春名　修介
【発明者】
【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
【氏名】　藤原　寛
【特許出願人】
【識別番号】　000005821
【氏名又は名称】　松下電器産業株式会社
【代理人】
【識別番号】　100097445
【弁理士】
【氏名又は名称】　岩橋　文雄

【選任した代理人】

【識別番号】　100103355

【弁理士】

【氏名又は名称】　坂口　智康

【選任した代理人】

【識別番号】　100109667

【弁理士】

【氏名又は名称】　内藤　浩樹

【手数料の表示】

【予納台帳番号】　011305

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9809938

【書類名】　明細書

【発明の名称】　コンパイラ、コンパイラを記録した記録媒体、コンパイル方法及びコンパイル装置

【特許請求の範囲】

【請求項１】　原始プログラムを目的プログラムに変換するコンパイラであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第１のサブプログラムから前記第２のサブプログラムへの移行ステップと、前記第２のサブプログラムから第１のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラ。

【請求項２】　原始プログラムを目的プログラムに変換するコンパイラであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第３のサブプログラムから呼び出される第４のサブプログラムを含む他のプログラムモジュールの記述に基づいて、

前記原始プログラムに含まれる第３のサブプログラムから第４のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存する

ステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第３のサブプログラムから前記第４のサブプログラムへの移行ステップと、前記第４のサブプログラムから第３のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換し、

或いはまた、前記原始プログラムに含まれる第３のサブプログラムから第４のサブプログラムを呼び出すステップを、

前記第３のサブプログラムから前記第４のサブプログラムへの移行ステップを含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラ。

【請求項３】　前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第３のサブプログラムから呼び出される第４のサブプログラムを含む他のプログラムモジュールの記述は、

前記第４のサブプログラム毎に記述される宣言である、請求項２に記載のコンパイラ。

【請求項４】　原始プログラムを目的プログラムに変換するコンパイラであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第５のサブプログラムから呼び出される第６のサブプログラムを含む他プログラム

モジュールに含まれる、呼び出された後のステップを、

前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的プログラムのステップに変換する、コンパイラ。

【請求項５】　原始プログラムを目的プログラムに変換するコンパイラであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第７のサブプログラムから呼び出される第８のサブプログラムを含む他プログラムモジュールの記述に基づいて、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、呼び出された後のステップを、

前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的プログラムのステップに変換し、

或いはまた、前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラ。

【請求項６】　前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第７のサブプログラムから呼び出される第８のサブプログラムを含む他のプログラムモジュールの記述は、

前記第８のサブプログラム毎に記述される宣言である、請求項５に記載のコンパイラ。

【請求項７】　前記呼び出される第２のサブプログラムまたは第４のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、

前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共用データメモリ領域を使用するのか、

または、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域を使用するのか、

に関する記述を含む、請求項１から請求項３のいずれか１項に記載のコンパイラ。

【請求項８】　前記他プログラムモジュール用の実行共有データメモリ領域を使用するのかまたは前期他プログラムモジュール用の実行固有データメモリ領域を使用するのかに関する記述は、

前記他のプログラムモジュールに含まれる外部変数毎に記述される宣言である、請求項７に記載のコンパイラ。

【請求項９】　前記原始プログラムに含まれる第１のサブプログラムまたは第３のサブプログラムから第２のサブプログラムまたは第４のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを保存するステップと、

前記呼び出される他のプログラムモジュール用の実行共有データメモリ領域アドレスを設定するステップと、

前記呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するステップと、

前記呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するステップと、

前記第１のサブプログラムまたは第３のサブプログラムから前記第２のサブプログラムまたは第４のサブプログラムへの移行ステップと、

前記第２のサブプログラムまたは第４のサブプログラムから前記第１のサブプログラムまたは第３のサブプログラムへの復帰後に、

前記保存した呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するステップと、

前記保存した呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するステップと、

を含む目的プログラムのステップに変換することがあり、

前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行固有データメモリ領域をアクセスするステップを、

前記設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、

前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行共有データメモリ領域をアクセスするステップを、

前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、

を含む目的プログラムのステップに変換する、請求項７または請求項８に記載のコンパイラ。

【請求項１０】　前記呼び出される第６のサブプログラムまたは第８のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、

前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域を使用するのか、

または、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プ

ログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域を使用するのか、

に関する記述を含む、請求項４から請求項６のいずれか１項に記載のコンパイラ。

【請求項１１】　前記他プログラムモジュール用の実行共有データメモリ領域を使用するのかまたは前記他プログラムモジュール用の実行固有データメモリ領域を使用するのかに関する記述は、

前記他のプログラムモジュールに含まれる外部変数毎に記述される宣言である、請求項１０に記載のコンパイラ。

【請求項１２】　前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、呼び出された後のステップを、

前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するステップと、

前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するステップに変換し、

前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行固有データメモリ領域をアクセスするステップを、

前記設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、

前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行共有データメモリ領域をアクセスするステップを、

前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、

を含む前記目的のプログラムのステップに変換する、請求項１０または請求項１１に記載のコンパイラ。

【請求項１３】　前記呼び出し元プログラムモジュール用の実行共有、および、実行固有データメモリ領域アドレスを保存するステップは、

前記第１のサブプログラムまたは第３のサブプログラムから第２のサブプログラムまたは第４のサブプログラムへ移行するために必要なデータよりも先に前記呼び出し元プログラムモジュール用の実行共有、および、実行固有データメモリ領域アドレスをスタックメモリに保存するステップを含む前記目的プログラムのステップである、請求項１から請求項３または請求項７から請求項９のいずれか１項に記載のコンパイラ。

【請求項１４】　前記原始プログラムに含まれる前記呼び出し元プログラムモジュールと、

前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムまたは第３のサブプログラムまたは第５のサブプログラムまたは第７のサブプログラムから呼び出される第２のサブプログラムまたは第４のサブプログラムまたは第６のサブプログラムまたは第８のサブプログラムを含む他プログラムモジュールとは、同じプログラムモジュールである、請求項１から請求項１３のいずれか１項に記載のコンパイラ。

【請求項１５】　コンピュータ読み取り可能な記録媒体であって、

請求項１から請求項１４のいずれか１項に記載のコンパイラを記録した記録媒体。

【請求項１６】　請求項１から請求項１４のいずれか１項に記載のコンパイラによって生成される目的プログラム。

【請求項１７】　コンピュータ読み取り可能な記録媒体であって、請求項１６に記載の目的プログラムを記録した記録媒体。

【請求項１８】　原始プログラムを目的プログラムに変換するコンパイル方法であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存する

ステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第１のサブプログラムから前記第２のサブプログラムへの移行ステップと、前記第２のサブプログラムから第１のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイル方法。

【請求項１９】　原始プログラムを目的プログラムに変換するコンパイル装置であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第１のサブプログラムから前記第２のサブプログラムへの移行ステップと、前記第２のサブプログラムから第１のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイル装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本特許出願に係る発明（以後単に「本発明」とも言う）は、あるプログラム言語で記述された原始プログラム（以後、「ソースプログラム」または「ソースプログラムコード」または「ソースコード」とも言う）を、あるコンピュータによって実行するための目的プログラム「以後、オブジェクトプログラム」または「オブジェクトコードプログラム」または「オブジェクトコード」とも言う）に変換（以後場合によっては「翻訳」と言うこともあるが同一内容を示す）するプログラムであるコンパイラに関するものである。

【０００２】

また本特許出願の明細書で言う、目的プログラムまたはオブジェクトプログラム等には、コンピュータが直接実行可能なプログラムコードに加え、コンピュータが直接実行可能なプログラムコードに変換する１段階である中間プログラムコードをも含むものであり、またそれに加え特に、コンピュータが直接実行可能なプログラムコードでは確定するべき実際にアクセスされるメモリアドレスが確定していない段階の中間コードプログラムをも含むものであり、更にまたそれに加え特に、コンピュータプラグラムが直接実行可能なプログラムコード全体を一体化する以前の段階の中間コードプログラムをも含むものである。

【０００３】

また例えば、いわゆる機器組み込みプログラムのようにオブジェクトプログラムを格納することのできるメモリ領域が限られており、しかも近年の携帯情報機器のようにその限られたメモリ領域に高度且つ高機能な情報処理のためのオブジェクトプログラムを格納しなければならないような目的に適う、オブジェクトプログラムを生成するためのコンパイラに関するものである。

【０００４】

更にまた、このような目的に適うオブジェクトプログラムとして再入可能（以後、「リエントラント：ｒｅｅｎｔｒａｎｔ」と言うこともある）なオブジェクトプログラムを自動生成するためのコンパイラに関するものである。

【０００５】

【従来の技術】

従来、時分割または優先度プリエンプティブタスクまたは、プロセススケジューリング機能、及びそれに類する機能を有するオペレーティングシステムや、関数型プログラミング言語を用いることを条件として制作されるプログラムであって、複数のタスクまたはプロセスからある一定の時間間隔の中で、同時に呼び出される可能性がある関数プログラムは再入可能であることが要求された。

【０００６】

そしてこのような再入可能関数プログラムは、人間のプログラマによってコーディングされ、再入可能性が満足されているか否かを自動的に判定するためのプログラムが作成されていた（例えば、特許文献１参照）。

【０００７】

この再入可能性を満足する条件は、判定を行う対象である関数を記述しているソースプログラムの全体を構文解析してその全フローを抽出し、その全フローでアクセスされるすべてのリソースに対して、スコープを決定し、そのリソースのスコープの種類に応じて完全な排他的制御が行われているか否かに依存するものであった。

【０００８】

【特許文献１】

特開２００１－１３４４３１号公報

【０００９】

【発明が解決しようとする課題】

しかしながら、ソースプログラムの全体を構文解析してその全フローを抽出し、その全フローでアクセスされるすべてのリソースに対して、スコープを決定し、そのリソースのスコープの種類に応じて完全な排他的制御が行われているか否かを完全に行うことは時間的な制約やコンピュータの処理能力の限界から、現実には非常に困難で、ある一定レベルの判定に留まらざるを得なかった。

【００１０】

そして従来の再入可能なプログラムの作成は、その本質的部分を人間のプログラマの作業によって記述されるため、多くの時間を要しながら不完全な排他的制

御しか実現することができず、その結果、プログラムを実行させた時に多くの誤動作を引き起こす原因となっていた。

【００１１】

またこのような再入プログラムはマルチタスクや時分割環境下で使用されることが多いため、その原因発見は極めて困難であった。

【００１２】

更にまた、このような再入可能なプログラムの更新や改訂を行うことは、更に一層困難なものであった。

【００１３】

本特許出願に係る発明のコンパイラ、コンパイラを記録した記録媒体、プログラム生成方法及びプログラム生成装置は、これらの課題を解決するため、自動的に再入可能な目的プログラムを生成することを目的とするものである。

【００１４】

【課題を解決するための手段】

上記課題を解決するため、本特許出願に係る請求項１に記載の発明は、原始プログラムを目的プログラムに変換するコンパイラであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第１のサブプログラムから前記第２のサブプログラムへの移行ステップと、前記第２のサブプログラムから第１のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラである。

【００１５】

本特許出願に係る請求項２に記載の発明は、原始プログラムを目的プログラムに変換するコンパイラであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第３のサブプログラムから呼び出される第４のサブプログラムを含む他のプログラムモジュールの記述に基づいて、前記原始プログラムに含まれる第３のサブプログラムから第４のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第３のサブプログラムから前記第４のサブプログラムへの移行ステップと、前記第４のサブプログラムから第３のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換し、或いはまた、前記原始プログラムに含まれる第３のサブプログラムから第４のサブプログラムを呼び出すステップを、前記第３のサブプログラムから前記第４のサブプログラムへの移行ステップを含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラである。

【００１６】

本特許出願に係る請求項３に記載の発明は、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第３のサブプログラムから呼び出される第４のサブプログラムを含む他のプログラムモジュールの記述は、前記第４のサブプログラム毎に記述される宣言である、請求項２に記載のコンパイラである

【００１７】

本特許出願に係る請求項４に記載の発明は、原始プログラムを目的プログラムに変換するコンパイラであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第５のサブプログラムから呼び出される第６のサブプログラムを含む他プログラムモジュールに含まれる、呼び出された後のステップを、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的プログラムのステップに変換する、コンパイラである。

【００１８】

本特許出願に係る請求項５に記載の発明は、原始プログラムを目的プログラムに変換するコンパイラであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第７のサブプログラムから呼び出される第８のサブプログラムを含む他プログラムモジュールの記述に基づいて、前記原始プログラムに含まれる他プログラムモジュールに含まれる、呼び出された後のステップを、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的プログラムのステップに変換し、或いはまた、前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスす

るステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラである。

【００１９】

本特許出願に係る請求項６に記載の発明は、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第７のサブプログラムから呼び出される第８のサブプログラムを含む他のプログラムモジュールの記述は、前記第８のサブプログラム毎に記述される宣言である、請求項５に記載のコンパイラである。

【００２０】

本特許出願に係る請求項７に記載の発明は、前記呼び出される第２のサブプログラムまたは第４のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共用データメモリ領域を使用するのか、または、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域を使用するのか、に関する記述を含む、請求項１から請求項３のいずれか１項に記載のコンパイラである。

【００２１】

本特許出願に係る請求項８に記載の発明は、前記他プログラムモジュール用の実行共有データメモリ領域を使用するのかまたは前期他プログラムモジュール用の実行固有データメモリ領域を使用するのかに関する記述は、前記他のプログラムモジュールに含まれる外部変数毎に記述される宣言である、請求項７に記載のコンパイラである。

【００２２】

本特許出願に係る請求項９に記載の発明は、前記原始プログラムに含まれる第１のサブプログラムまたは第３のサブプログラムから第２のサブプログラムまたは第４のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジ

ュール用の実行共有データメモリ領域アドレスを保存するステップと、前記呼び出される他のプログラムモジュール用の実行共有データメモリ領域アドレスを設定するステップと、前記呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するステップと、前記第１のサブプログラムまたは第３のサブプログラムから前記第２のサブプログラムまたは第４のサブプログラムへの移行ステップと、前記第２のサブプログラムまたは第４のサブプログラムから前記第１のサブプログラムまたは第３のサブプログラムへの復帰後に、前記保存した呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するステップと、前記保存した呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するステップと、を含む目的プログラムのステップに変換することがあり、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行固有データメモリ領域をアクセスするステップを、前記設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行共有データメモリ領域をアクセスするステップを、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む目的プログラムのステップに変換する、請求項７または請求項８のいずれかに記載のコンパイラである。

【００２３】

本特許出願に係る請求項１０に記載の発明は、前記呼び出される第６のサブプログラムまたは第８のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域を使用するのか、または、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュール

の実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域を使用するのか、に関する記述を含む、請求項４から請求項６のいずれか１項に記載のコンパイラである。

【００２４】

本特許出願に係る請求項１１に記載の発明は、前記他プログラムモジュール用の実行共有データメモリ領域を使用するのかまたは前記他プログラムモジュール用の実行固有データメモリ領域を使用するのかに関する記述は、前記他のプログラムモジュールに含まれる外部変数毎に記述される宣言である、請求項１０に記載のコンパイラである。

【００２５】

本特許出願に係る請求項１２に記載の発明は、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、呼び出された後のステップを、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するステップと、前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するステップに変換し、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行固有データメモリ領域をアクセスするステップを、前記設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行共有データメモリ領域をアクセスするステップを、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的のプログラムのステップに変換する、請求項１０または請求項１１に記載のコンパイラである。

【００２６】

本特許出願に係る請求項１３に記載の発明は、前記呼び出し元プログラムモジ

ュール用の実行共有、および、実行固有データメモリ領域アドレスを保存するステップは、前記第１のサブプログラムまたは第３のサブプログラムから第２のサブプログラムまたは第４のサブプログラムへ移行するために必要なデータよりも先に前記呼び出し元プログラムモジュール用の実行共有、および、実行固有データメモリ領域アドレスをスタックメモリに保存するステップを含む前記目的プログラムのステップである、請求項１から請求項３または請求項７から請求項９のいずれか１項に記載のコンパイラである。

【００２７】

本特許出願に係る請求項１４に記載の発明は、前記原始プログラムに含まれる前記呼び出し元プログラムモジュールと、前記呼び出し元プログラムモジュールに含まれる第１のサブプログラムまたは第３のサブプログラムまたは第５のサブプログラムまたは第７のサブプログラムから呼び出される第２のサブプログラムまたは第４のサブプログラムまたは第６のサブプログラムまたは第８のサブプログラムを含む他プログラムモジュールとは、同じプログラムモジュールである、請求項１から請求項１３のいずれか１項に記載のコンパイラである。

【００２８】

本特許出願に係る請求項１５に記載の発明は、コンピュータ読み取り可能な記録媒体であって、請求項１から請求項１４のいずれか１項に記載のコンパイラを記録した記録媒体である。

【００２９】

本特許出願に係る請求項１６に記載の発明は、請求項１から請求項１４のいずれか１項に記載のコンパイラによって生成される目的プログラムである。

【００３０】

本特許出願に係る請求項１７に記載の発明は、コンピュータ読み取り可能な記録媒体であって、請求項１６に記載の目的プログラムを記録した記録媒体である。

【００３１】

本特許出願に係る請求項１８に記載の発明は、原始プログラムを目的プログラムに変換するコンパイル方法であって、前記原始プログラムに含まれる呼び出し

元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第１のサブプログラムから前記第２のサブプログラムへの移行ステップと、前記第２のサブプログラムから第１のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイル方法である。

【００３２】

本特許出願に係る請求項１９に記載の発明は、原始プログラムを目的プログラムに変換するコンパイル装置であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第１のサブプログラムから他プログラムモジュールに含まれる第２のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第１のサブプログラムから前記第２のサブプログラムへの移行ステップと、前記第２のサブプログラムから第１のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイル装置である。

【００３３】

【発明の実施の形態】

本特許出願に係る発明（以後、「本発明」とも言う）の実施の形態について図を参照して説明する。

【００３４】

（実施の形態１）

図１は本発明の第１の実施の形態であるコンパイラが変換するソースプログラムとそのコンパイラによって変換されたオブジェクトプログラムの構造を示す図である。

【００３５】

但し、これらのプログラムは必ずしも１つにまとまった形で存在するとは限らず、幾つかの物理的な媒体、ファイル等に分散して記録されていることもある。

【００３６】

これらのプログラムは、アプリケーションプログラムモジュール１２１とアプリケーションプログラムモジュール１３１とライブラリプログラムモジュール１５１と、その他本発明の説明に直接関係しないので図示していない幾つかのプログラムモジュールとを含んでいる。

【００３７】

プログラムモジュールとは、プログラムを何らかの単位（例えば、機能単位、リンク単位、バージョンアップ単位等）で分割したものであり、プログラムコード領域と固有データ領域を保持する。尚ここでは固有のデータ領域としたが、必ずしも厳密に固有であることは必要でなく、他の何らかの目的や用途と兼用しているものであっても良い。このことは他の説明についても同様である。

【００３８】

プログラムモジュールのプログラムコード領域とは、そのプログラムモジュール内に記述されたプログラムの実行命令等を保持する領域である。

【００３９】

プログラムモジュールの固有データ領域とは、そのプログラムモジュール内で定義された、外部変数、および静的変数の実体を保持する領域である。

【００４０】

以後、外部変数と静的変数の両方を含めて、外部変数と言うこともある。

【００４１】

アプリケーションプログラムモジュール１２１は、プログラムコード領域１２３とこのアプリケーションプログラムモジュール１２１の固有データ領域１２５と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【００４２】

アプリケーションプログラムモジュール１３１は、プログラムコード領域１３３とこのアプリケーションプログラムモジュール１３１の固有データ領域１３５と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【００４３】

これらのプログラムはアプリケーションプログラムモジュール１２１とアプリケーションプログラムモジュール１３１は、呼び出し元プログラムであって、共通にライブラリプログラムモジュール１５１を呼び出す。この呼び出しは通常、関数コール、サブルーチンコール等の形態で行われる。

【００４４】

それが図１に示すプログラム呼び出し１とプログラム呼び出し２（再入：ｒｅｅｎｔｒｙ）である。

【００４５】

これらプログラムモジュールは通常、幾つかのサブプログラムを含んでいる。このサブプログラムは通常、関数やサブルーチンやその他である。また、これら関数やサブルーチン１つをプログラムモジュールと呼ぶこともある。

【００４６】

また通常、プログラムモジュールの呼び出しは、他プログラムモジュールのサブプログラムを呼び出すことによって行うが、同一プログラムモジュール内のサブプログラムを呼び出すこともある。

【００４７】

いわゆるマルチタスク環境、その他時分割的に複数のプロセスまたはタスクまたはアプリケーションが並行して実行される、或いは実行しなければならない環境では、例えば、アプリケーションプログラムモジュール１２１がライブラリプログラムモジュール１５１を呼び出して実行中に、並行して、アプリケーションプログラムモジュール１３１がライブラリプログラムモジュール１５１を呼び出して実行することが発生する。

【００４８】

このような状態は種々の情報機器で無数に近く発生するが、非常に簡単な幾つかの例を示す。

【００４９】

例えば、携帯電話等のモバイル情報通信機能を有する機器では、ユーザーが送信するためのデータを入力している時にも、同時に並行して着信の有無を監視していなければならない。

【００５０】

このデータ入力と着信の監視を別のＣＰＵで行うこともできるが、ＣＰＵの節約のため１ＣＰＵで行おうとするならば、時分割的に並行して行う必要が起こる。

【００５１】

そしてある基本的な関数型サブルーチンが、これらデータ入力アプリケーションと着信監視アプリケーションとの両方で使用される関数型サブルーチンであったならば、この関数型サブルーチンは、一方のアプリケーションから呼び出されて実行中に、もう一方のアプリケーションから呼び出されることが起こり得る。

【００５２】

このように複数のタスク等が時間的に重複して同一のプログラムモジュール（関数またはサブプログラムまたはサブルーチン等）を呼び出し、実行するようなケースを「再入：ｒｅｅｎｔｒｙ」と言い、あるプログラムモジュールが時間的に重複・並行して複数のタスクを実行中のプログラムから呼び出され、実行され得るように構成されていることを「再入可能：ｒｅｅｎｔｒａｎｔ」と言う。

【００５３】

従来から、プログラムモジュールを再入可能であるように構成する方法は幾つか知られている。

【００５４】

その第１は、いわゆる外部変数を使用せず、全て内部変数（以後、Ｌｏｃａｌ変数と言うこともあるが同一内容を表す）で処理することであった。内部変数はこのプログラムモジュールが呼び出される毎にスタック領域に新たに確保され、このプログラムモジュールの処理が終了するのと同時に解放されるため、全て内部変数を使用している限り、変数へのアクセスが衝突することは発生せず、同時に並行的に実行されても問題はなかった。

【００５５】

その第２は、いわゆる外部変数を使用するが、この外部変数に対するアクセスをいわゆるセマフォ（ｓｅｍａｐｈｏｒｅ：腕木信号機）と呼ばれる状態変数によって調停する方法である。

【００５６】

第１の方法では、外部変数を使用することができないため、このプログラム以外のプログラムモジュールとデータを受け渡しするのに制約を受けることがある。

【００５７】

第２の方法では、煩雑なセマフォに対するテスト・アンド・セットを外部変数にアクセスする毎に行わなければならず、多くの負担をプログラマに課し、間違いや誤動作の原因となることが多かった。

【００５８】

ライブラリプログラムモジュール１５１は、プログラムコード領域１５３とこのライブラリプログラムモジュール１５１の固有データ領域１５５、１５７と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【００５９】

アプリケーションプログラムモジュール１２１用データ領域１５５とアプリケーションプログラムモジュール１３１用データ領域１５７とは、このライブラリ

プログラムモジュール１５１を呼び出した元のプログラムモジュールによって区別されている。

【００６０】

例えば、アプリケーションプログラムモジュール１２１からこのライブラリプログラムモジュール１５１が呼び出された時には、プログラムコード領域１５３は外部変数領域としてアプリケーションプログラムモジュール１２１用データ領域１５５をアクセスしてその処理を行い、アプリケーションプログラムモジュール１３１からこのライブラリプログラムモジュール１５１が呼び出された時には、プログラムコード領域１５３は外部変数領域としてアプリケーションプログラムモジュール１３１用データ領域１５７をアクセスしてその処理を行う。

【００６１】

このように呼び出し元プログラム単位に、ライブラリプログラムモジュールが使用する固有データ領域、特に外部変数のために使用するデータ領域を区分して使用することによって、外部変数に対するアクセスが衝突することを気にせず、プログラムを作成することができる。

【００６２】

既に説明したように、内部変数については、このライブラリプログラムモジュール１５１が呼び出される毎に、そのために領域がスタックに確保されるため、アクセスが衝突する問題は生じない。この状態の例を図４（ｂ）のＬｏｃａｌ変数１～３に示す。

【００６３】

尚ここで、データ領域を呼び出し元アプリケーションプログラム単位に用意して使用すると説明したが、ここで言うアプリケーションプログラムを、アプリケーションタスク単位と読み替えても良く、ここで言うアプリケーションプログラムはアプリケーションタスクと同じ概念であると考えても良い。

【００６４】

また、ここで言うアプリケーションプログラムを、アプリケーションプロセス単位と読み替えても良く、ここで言うアプリケーションプログラムはアプリケーションプロセスと同じ概念であると考えても良い。

【００６５】

より詳しくアプリケーションプログラムモジュール１２１がライブラリプログラムモジュール１５１を呼び出す時の処理手順を図２と図３に示す。

【００６６】

図３（ａ）、（ｂ）に示すソースプログラムは、Ｃ言語を例として記述している。

【００６７】

アプリケーションプログラムモジュール１２１のソースプログラムに記述された「ｂａｒ（Ａ，Ｂ）；」命令がオブジェクトプログラムに変換されると、ライブラリプログラムモジュール１５１に含まれる関数型サブルーチンであるｂａｒ（以後、関数型サブプログラムｂａｒと言うこともあり、単に関数ｂａｒと言うこともあるが同一である）を呼び出すオブジェクトプログラムが生成される（図３（ａ）の１～７）。

【００６８】

アプリケーションプログラムモジュール１２１のプログラムコード領域１２３に含まれるプログラムは、ライブラリプログラムモジュール１５１に含まれる関数ｂａｒを呼び出すに際し、まず現在実行中のアプリケーションプログラムモジュール１２１が使用しているデータ領域１２５の先頭アドレスである０ｘ０６００（１６進数表記で０６００であり、１０進数表記に変換すると、$0*16^3+6*16^2+0*16+0$である、以後同じ）を保存しているＤＰレジスタ（以後単にＤＰと言うこともある）の内容をスタックに保存する（図２（ａ）の１、図３（ａ）の１））。

【００６９】

これはライブラリプログラムモジュール１５１から戻ってきた時に、アプリケーションプログラムモジュール１２１がＤＰの内容を再度０ｘ０６００に設定し、アプリケーションプログラムモジュール１２１が自分自身のための固有データ領域１２５を再度使用することができるようにするためである。尚ここでは固有のデータ領域としたが、必ずしも厳密に固有であることは必ずしも必要でなく、他の何らかの用途や目的、その他と共有であっても良い。このことは他の説明に

ついても同様である。

【００７０】

ＤＰレジスタは、実行中のプログラムモジュールのデータ領域の先頭アドレスを保持する領域である。この領域として、通常特別なレジスタを用意するが、必ずしも特別なレジスタを用意せず、汎用的なレジスタを使用してもよい。

【００７１】

次に、アプリケーションプログラムモジュール１２１に含まれるプログラムは、ライブラリプログラムモジュール１５１に引き渡すデータである引数Ａと引数Ｂをスタックに保存する（図２（ａ）の２、図３（ａ）の２）。

【００７２】

この時のスタックの状態を図４（ａ）に示す。

【００７３】

図４（ａ）に示すようにスタック（以後、スタックメモリと言うこともある）はＬａｓｔ－ｉｎ／Ｆｉｒｓｔ－ｏｕｔ型のメモリ領域で、あるプログラムモジュールから他のプログラムモジュール（例えば、関数やサブルーチン、サブプログラム、その他である）を呼び出す時に必要なデータを保存して、プログラムモジュール間でデータの引き渡しに使ったり、Ｌｏｃａｌ変数用メモリ領域として使用したり、呼び出された関数等から呼び出し元のプログラムに戻るためのアドレスの保存や、その他の目的に使用されるものである。

【００７４】

スタックのデータを出し入れする領域のアドレスは、スタックポインタと呼ばれるレジスタで示される（以後、単にスタックポインタ或いはスタックポインタレジスタと呼ぶこともある）。

【００７５】

このスタックの使われ方を例えば図４（ｂ）に示す。例えば図４（ｂ）に示すスタックの上から３番目に格納されているローカル変数２に対してアクセスするには、スタックポインタの値を変更するか或いは、スタックポインタレジスタの値マイナス２の値をアドレスとして間接参照を行うこともできる。

【００７６】

次に、アプリケーションプログラムモジュール１２１に含まれるプログラムは、自分自身の固有データ領域１２５内の領域に記憶している、関数ｂａｒを含むライブラリプログラムモジュール１５１で使用する、アプリケーションプログラムモジュール１２１固有のデータ領域１５５のアドレス（０ｘ８００）を読み出し、そのアドレスをＤＰに設定する（図２（ａ）の３、図３（ａ）の３）。

【００７７】

前記関数ｂａｒを含むライブラリプログラムモジュール１５１で使用する、アプリケーションプログラムモジュール１２１固有のデータ領域１５５のアドレス（０ｘ８００）を保持する領域は、通常コンパイラが設定する。しかし必ずしも、前記領域をコンパイラが設定せず、ライブラリとして用意し、リンク時に設定してもよい。

【００７８】

次に、アプリケーションプログラムモジュール１２１に含まれるプログラムは、呼び出し先プログラムである関数ｂａｒから自分自身に帰ってくる時のアドレス、即ち関数ｂａｒへのジャンプ命令の次の命令のアドレスをスタックに保存し、関数ｂａｒへジャンプする（図２（ａ）の４、図３（ａ）の４）。

【００７９】

図２（ｂ）に示すように、呼び出されるプログラムである関数ｂａｒは、ライブラリプログラムモジュール１５１に含まれるプログラムコード領域１５３に含まれている。

【００８０】

この関数ｂａｒは引数としてスタックに保存された引数Ａと引数Ｂとを取り出して処理に使用する。この時のスタックの状態は前記図４（ａ）に示す通りである。

【００８１】

更にこの関数ｂａｒは外部変数として、例えばこのライブラリプログラムモジュール１５１で定義された外部変数Ｃを使用する（図２（ｂ）の５、図３（ｂ）の５）。

【００８２】

関数ｂａｒではこの外部変数Ｃに対してアクセスし、例えばそのメモリ領域に値１０を代入するに際し、ＤＰに保存されている値（０ｘ８００）にＤＰからの相対アドレス（０ｘ５０）を加算したアドレス（０ｘ８５０）に値１０を代入する。この命令形式を図２（ｂ）の５と図３（ｂ）の５に示す。

【００８３】

これによって関数ｂａｒは、アドレス０ｘ８００から始まるメモリ領域であるアプリケーションプログラムモジュール１２１用の固有データ領域１５５に対してアクセスすることができる（図２（ｂ）の５）。

【００８４】

関数ｂａｒでの処理が終ると、プログラムの制御は元のアプリケーションプログラムモジュール１２１に含まれるプログラムに戻される。

【００８５】

関数ｂａｒからの復帰直後に、スタックから引数Ａ、Ｂが破棄される（図２（ａ）の６、図３（ａ）の６）。尚ここでは関数ｂａｒからの復帰直後としたが、必ずしも厳密に直後であることは必要でなく、何らかの処理の後であっても良い。このことは他の説明についても同様である。

【００８６】

次に、スタックに保存していた元のＤＰの値、この例では０ｘ６００がスタックから取り出されてＤＰに戻される（図２（ａ）の７、図３（ａ）の７）。

【００８７】

これによって呼び出し元のアプリケーションプログラムモジュール１２１に含まれるプログラムは、再度自分自身の固有データ領域１２５に対してアクセスすることが可能になる（図２（ａ））。

【００８８】

尚この時、図４に示す通り、呼び出し元のＤＰの値は、スタックの一番下に保存されているので、呼び出されたライブラリプログラムモジュール１５１の側ではその内容を読み出す必要がないと共に、実際に読み出さなくても他の必要な処理を全て行うことが可能であり、不要なアクセスを行う必要が生じない。

【００８９】

また例えば次の第２の実施の形態に示すような他の実施の形態であって、ＤＰをスタックに保存するか保存しないか選択ができる実施の形態では、ＤＰをスタックの一番下に保存することによって、ＤＰがスタックに保存されているか保存されていないかに係わらず、同一アドレスに対するアクセスによって同じ処理を行うことができ、ＤＰがスタックに保存されているか否かを判断して処理を分ける必要がない。

【００９０】

即ち、ＤＰをスタックの一番下に保存することによって、ＤＰがスタックに保存されているか保存されていないかに係わらず、呼び出されたライブラリプログラムモジュール１５１の側で必要なデータ、例えば引数や帰りアドレスを取り出すスタック位置は変わらない。

【００９１】

即ち、引数や帰りアドレスを取り出すスタックのスタックポインタからの相対位置は変化しないので、それを考慮してこれら引数や帰りアドレスを取り出すことは必要でない。

【００９２】

尚、この例では、アプリケーションプログラムモジュールから、ライブラリプログラムモジュールを呼び出す例について書いたが、ライブラリプログラムモジュールから、他のライブラリモジュールを呼び出すことも可能であり、その際の処理は、アプリケーションプログラムモジュールから、他ライブラリモジュールを呼び出す場合の処理と同じであるから、説明を省略する。

【００９３】

尚、この例では、他のプログラムモジュールを呼び出す例について書いたが、同一プログラムモジュールを呼び出すことも可能であり、その際の処理は、他のプログラムモジュールを呼び出す場合の処理と同じであるから、説明を省略する。

【００９４】

（実施の形態２）

前記本発明の第１の実施の形態では、アプリケーションプログラムモジュール

１２１のソースプログラムに記述された「ｂａｒ（Ａ，Ｂ）；」命令がオブジェクトプログラムに変換されることによって生成されるオブジェクトプログラムは、現在実行中のアプリケーションプログラムモジュール１２１が使用しているデータ領域１２５の先頭アドレスを保存しているＤＰレジスタの内容をスタックに保存し、ライブラリプログラムモジュール１５１に引き渡すデータである引数Ａと引数Ｂをスタックに保存し、呼び出される関数ｂａｒで使用する、アプリケーションプログラムモジュール１２１固有のデータ領域１５５のアドレスを読み出してそのアドレスをＤＰに設定し、関数ｂａｒから自分自身に帰ってくる時のアドレスをスタックに保存し、関数ｂａｒへジャンプし、関数ｂａｒからの復帰時には、引数Ａと引数Ｂをスタックから破棄し、スタックに保存している元のＤＰの値を再設定するステップを含むものであった。

【００９５】

次に説明する本発明の第２の実施の形態では、上記のような処理を行うか、それとも上記の処理の一部を行わずに省略するかの記述を、ライブラリプログラムモジュール１５１のソースプログラムの記述は含むものである。そのための記述を図８に示す。

【００９６】

本発明の第１の実施の形態では、他プログラムモジュールの呼び出しを例に挙げ説明したが、本発明の第２の実施の形態では、同一モジュール内の関数ｂａｒの呼び出しを例に挙げて説明する。

【００９７】

図８ではＣ言語を例に記述している。この例ではＣ言語の言語仕様を拡張して、新規キーワードとして「ｐｒｉｖａｔｅ」を追加し、ソースコード上で「ｐｒｉｖａｔｅ」修飾詞をつけることにより、宣言を記述している。

【００９８】

例えば図８に示す関数ｂａｒの呼び出しでは、上記本発明の第１の実施の形態で行った処理の一部を行うことなく、省略できることを示す。

【００９９】

そのための宣言が、例えば図８に示す関数ｂａｒでは、「ｐｒｉｖａｔｅ　ｖ

ｏｉｄ　ｂａｒ（ｉｎｔ　Ａ，ｉｎｔＢ）」である。

【０１００】

図８に示すように、ライブラリモジュールプログラムのソースプログラムに含まれる関数ｂａｒには、関数ｂａｒが「ｐｒｉｖａｔｅ」である旨の記述が含まれている。

【０１０１】

この記述は「関数ｂａｒがいわゆるプライベイト（ｐｒｉｖａｔｅ）関数である」旨を宣言する記述であり、例えば「プライベイト（ｐｒｉｖａｔｅ）宣言」或いは「プライベイト（ｐｒｉｖａｔｅ）関数宣言」と呼ばれることもある。

【０１０２】

このプライベイト関数ｂａｒを呼び出す記述が行われると、第１の実施の形態で説明したような、呼び出し元であるプログラムモジュールが使用しているＤＰの保存や、呼び出されるプログラムモジュールで使用される固有データ領域アドレスを示すＤＰの設定や、関数ｂａｒからの復帰時に、スタックに保存している元のＤＰの値の再設定を行うためのオブジェクトプログラムのステップは生成されない。

【０１０３】

このプライベイト関数ｂａｒを呼び出す記述がライブラリプログラムモジュール１５１のプログラムコード領域１５３で行われると、図７に示すように、関数ｂａｒで使用するためのデータである引数Ａや引数Ｂを関数ｂａｒに引き渡すためにスタックへ保存するステップと、関数ｂａｒへジャンプする命令の次の命令のアドレスをスタックへ格納し、関数ｂａｒへジャンプするステップと、関数ｂａｒからの復帰時に、引数Ａや引数Ｂをスタックから破棄するステップと、を含むオブジェクトプログラムが本実施の形態のコンパイラによって生成される。

【０１０４】

これによって、必ずしも呼び出し元であるプログラムモジュールと呼び出されるプログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、関数ｂａｒをプライベイト宣言することによって、ＤＰの保存や設定や再設定のステップを省略し、オブジェクトプログラムの実行を高速化することが

でき、不要なステップを出力しないことでオブジェクトプログラムサイズを小さくすることもできる。

【０１０５】

このようなプライベイト宣言された関数ｂａｒが実行されると、図７に示すように、ＤＰの内容は呼び出し元プログラムであるプログラムモジュールの状態のまま変化していないので、例えば外部変数Ｃに対してデータの読み出し等が実行されると、呼び出し元であるプログラムモジュールの固有データ領域（この場合、固有データ領域１５５）がアクセスされる。

【０１０６】

また、図示していないが、プライベイト宣言された関数ｂａｒから呼び出し元プログラムへリタンした時にも、ＤＰの再設定等を行う必要はない。

【０１０７】

またこれまで説明した実施の形態では、例えば関数ｂａｒの前に「ｐｒｉｖａｔｅ」と記述されるとプライベイト関数であることが宣言され、ＤＰの保存等の一連の処理が行われず、特に何も記述されなければプライベイト関数でなく、ＤＰの保存等の処理が行われる。

【０１０８】

しかし他の実施の形態では、例えば明示的にプライベイト関数でないことを示すために「ｐｕｂｌｉｃ」等の記述を行うことも可能である。

【０１０９】

このような記述は、「パブリック（ｐｕｂｌｉｃ）宣言」或いは「パブリック（ｐｕｂｌｉｃ）関数宣言」と呼ばれることもあり、この記述を行うことによって明示的にＤＰの保存等の一連の処理を行うことを示すことができる。

【０１１０】

また他の実施の形態では、例えばこの「プライベイト宣言」が行われる関数は、同一プログラムモジュール内でのみ呼び出される、或いは呼び出すことが可能な関数であっても良く、更に他の実施の形態では、この「プライベイト宣言」が行われる関数は、異なるプログラムモジュール間でのみ呼び出される、或いは呼び出すことが可能な関数であっても良い。

【０１１１】

尚、この例では新規キーワード「ｐｒｉｖａｔｅ」や「ｐｕｂｌｉｃ」を追加し、Ｃ言語を拡張して宣言を行っているが、プラグマ等を用いることにより、言語使用を拡張することなく、宣言を行ってもよい。

【０１１２】

また、別ファイルとして記述したり、コンパイラの引数として指定してもよい。

【０１１３】

（実施の形態３）

前記本発明の第１の実施の形態では、関数ｂａｒが使用する全ての外部変数（例えばＣ）に対してアクセスするに際し、ＤＰに保存されている値（例えば、０ｘ８００）に、ＤＰからの相対アドレス（例えば、０ｘ５０）を加算したアドレス（例えば、０ｘ８５０）でアクセスした。

【０１１４】

その結果、関数ｂａｒは、使用する全ての外部変数用領域として、アドレス例えば、０ｘ８００から始まるメモリ領域であるアプリケーションプログラムモジュール１２１用固有データ領域１５５を使用することができる。

【０１１５】

しかし次に説明する本発明の第３の実施の形態では、例えば関数ｂａｒを呼び出す複数のアプリケーションプログラムモジュールが共通にアクセスするデータメモリ領域に関する記述を、例えば関数ｂａｒの原始プログラムはその変数宣言部に含むことができる。

【０１１６】

その記述の例を図６示す。

【０１１７】

図６では、Ｃ言語を例として宣言を記述している。この例ではＣ言語の言語仕様を拡張して新規のキーワード「ｃｏｍｍｏｎ」を追加し、ソースコード上でこの「ｃｏｍｍｏｎ」修飾詞をつけることにより宣言を記述している。

【０１１８】

例えば図６に示す関数ｂａｒのプログラムでは、関数ｂａｒが使用する外部変数としてＤとＣを使用する（図６（ａ））。

【０１１９】

その宣言が例えば図６の「ｃｏｍｍｏｎ　ｉｎｔ　Ｄ；」と「ｉｎｔ　Ｃ；」の記述である（図６（ａ））。

【０１２０】

そして更に「ｃｏｍｍｏｎ　ｉｎｔ　Ｄ；」の記述は、この外部変数Ｄはこの関数ｂａｒの複数の呼び出し元アプリケーションプログラムが共通にアクセスする外部変数であることを宣言している（図６（ａ））。

【０１２１】

尚、この例では新規のキーワード「ｃｏｍｍｏｎ」を追加し、Ｃ言語を拡張して宣言を行っているが、プラグマ等を用いて言語仕様を拡張することなく宣言を行ってもよい。

【０１２２】

また、別ファイル等に記述したり、コンパイラの引数として指定することによって宣言を行ってもよい。

【０１２３】

図６に示すように、ライブラリモジュールプログラムのソースプログラムに含まれる外部変数Ｄには、この外部変数Ｄが、「ｃｏｍｍｏｎ」である旨の記述が含まれている。

【０１２４】

この記述は、「外部変数Ｄがいわゆる共通外部変数である」旨を宣言する記述であり、例えば「コモン（ｃｏｍｍｏｎ）宣言」或いは「コモン（ｃｏｍｍｏｎ）外部変数宣言」と呼ばれることもある。

【０１２５】

図５（ａ）は、プログラムコード領域５５３が、再入可能な関数ｂａｒに相当し、このプログラムから固有データ領域であるアプリケーションプログラムモジュール１２１用データ領域５５５と、共有データ領域である共有データ領域５５９との両方がアクセスされ得ることを示している（図５（ａ））。尚ここでは共

有データ領域としたが、必ずしも厳密に他の何らかの目的や用途と共有されていることは必要でなく、たまたま他の何らかの目的や用途と共有されていないものであってもかまわない。このことは他の説明についても同様である。

【０１２６】

図５（ｂ）はより具体的に、外部変数Ｃは固有データ領域であるアプリケーションプログラムモジュール１２１用データ領域５５５に格納され、外部変数Ｄは共有データ領域アクセスによってアクセスされる共有データ領域５５９に格納されることを示している（図５（ｂ））。

【０１２７】

図６の「ｉｎｔ　Ｃ；」の記述は、この外部変数Ｃは、この関数ｂａｒの複数の呼び出し元アプリケーションプログラム毎に固有のメモリ領域を使用することを宣言している記述であり、この記述によって、外部変数Ｃは固有データ領域アクセスによってアクセスされるアプリケーションプログラムモジュール（この例ではプログラムモジュール１２１）用の固有のデータ領域５５５に格納される（図５（ｂ））。

【０１２８】

そして関数ｂａｒでは、呼び出し元アプリケーションプログラム（例えばプログラムモジュール１２１）毎に固有のデータ領域５５５に格納されている外部変数Ｃに対してアクセスし、例えばそのメモリ領域に、値１０を代入するに際し、ＤＰに保存されている値（例えば、０ｘ８００）に、ＤＰからの相対アドレス（例えば、０ｘ５０）を加算したアドレス（例えば、０ｘ８５０）に、値１０を代入する。この命令形式を図５（ｂ）と図６に示す。

【０１２９】

この時の命令形式は前記第１の実施の形態と同一である。

【０１３０】

これによって関数ｂａｒは、例えばアドレス０ｘ８００から始まるメモリ領域であるアプリケーションプログラムモジュール１２１用の固有データ領域５５５に対してアクセスすることができる（図５（ｂ））。これによって関数ｂａｒに複数のアプリケーションプログラムモジュール（例えば、図１（ａ）のアプリケ

ーションプログラムモジュール１２１とアプリケーションプログラムモジュール１３１）から同時に呼び出され、その結果再入が起こっても外部変数へのアクセスが衝突することはない。

【０１３１】

図６の「ｃｏｍｍｏｎ　ｉｎｔ　Ｄ；」の記述は、この外部変数Ｄは、関数ｂａｒの複数の呼び出し元アプリケーションプログラムが共有するメモリ領域を使用することを宣言している記述であり、この記述によって、外部変数Ｄは共有データ領域アクセスによってアクセスされる複数のアプリケーションプログラムモジュールが共通に使用する共有データ領域５５９に格納される（図５（ｂ））。

【０１３２】

そして関数ｂａｒでは、呼び出し元アプリケーションプログラム（例えばプログラムモジュール１２１）が共通に使用する共有メモリ領域５５９に格納されている外部変数Ｄに対してアクセスし、例えばそのメモリ領域に、値２０を代入するに際し、特別なレジスタＳＤＰに保存されている値（例えば、０ｘＦ００）に、ＳＤＰからの相対アドレス（例えば、０ｘ６０）を加算したアドレス（例えば、０ｘＦ６０）に、値２０を代入する。この命令形式を図５（ｂ）、図６に示す。

【０１３３】

このＳＤＰは特別なレジスタであり、全アプリケーションプログラムに対して１つの共有データ領域の先頭アドレスを保存している。またこのＳＤＰに保存されている値である全アプリケーションプログラムに対して１つの共有データ領域の先頭アドレス自体をＳＤＰと呼ぶこともある。

【０１３４】

或いは他の実施の形態ではこのＳＤＰとして必ずしも特別なレジスタを使用せず、汎用的なレジスタを使用することもできる。このような実施の形態は汎用のレジスタを多く備えているようなＭＰＵにおいて特に有効であり、更にこの汎用レジスタがメモリアクセスのためのオフセットアドレスとして使用できるＭＰＵでは更に有効である。

【０１３５】

或いはまた他の実施の形態ではこのＳＤＰとして必ずしも特別なレジスタや汎用のレジスタを使用せず、通常のメモリやスタックを使用することもできる。

【０１３６】

しかしこのような実施の形態ではこのスタックや通常のメモリに格納されたＳＤＰをオフセットアドレスとして使用してメモリアクセスするために、幾らかのステップが必要になることもある。

【０１３７】

例えばそのステップとは、スタックやメモリに格納されているＳＤＰを読み出し、メモリアクセスのためのオフセットアドレスとして使用できるレジスタに格納したり、そのための準備として現在のレジスタの内容を退避するステップ等である。

【０１３８】

これらのような方法で、このＳＤＰレジスタが記憶している値をオフセットアドレスとして使用してメモリ領域アクセスすることによって、関数ｂａｒは、例えばアドレス０ｘＦ００から始まるメモリ領域であるアプリケーションプログラムモジュール共有データ領域５５９に対してアクセスすることができる（図５（ｂ））。

【０１３９】

ＳＤＰの値は、図示していないが例えば図２（ａ）のデータ領域１２５の、ライブラリプログラムモジュール１５１の自分固有データ領域アドレスと同様にデータ領域１２５に記憶しておき、アプリケーションプログラムモジュール１２１からライブラリプログラムモジュール１５１を呼び出す時、その呼び出しステップの中で保存と設定を行うこともできる。

【０１４０】

例えば、図２（ａ）で「１．現在のＤＰ（０ｘ０６００）をスタックに保存」するステップの前のステップとして、「（０．５）　現在のＳＤＰをスタックに保存」するステップを含めることも可能である。

【０１４１】

また図２（ａ）で「３．新しいＤＰ（０ｘ０８８０）を設定」するステップの

前のステップとして、「（２．５）　新しいＳＤＰ（０ｘ０Ｆ００）を設定」するステップを含めることも可能である。

【０１４２】

このように現在のＳＤＰをスタックに保存するステップを、引数Ａや引数Ｂをスタックに保存するステップよりも先に行うことによって、呼び出されるライブラリプログラムモジュールの側でこのスタックに保存されたＳＤＰの有無に関与しなくてもよいことは、前記第１の実施の形態と同じであるから詳細な説明を省略する。

【０１４３】

このような実施の形態によって、関数ｂａｒが、複数のアプリケーションプログラムモジュール（例えば、図１（ａ）のアプリケーションプログラムモジュール１２１とアプリケーションプログラムモジュール１３１）から同時に呼び出され、その結果再入が起こった時、明示的に、その外部変数の幾つかを共通に使用し、データの共有、受け渡し、その他の処理を行うことができる。

【０１４４】

またこの時、前記したセマフォ等を使った調停が行われることがある、或いは行わなければならないことも発生し得るが、本発明の内容と直接関係しないので説明を省略する。

【０１４５】

（実施の形態４）

図９に本発明の第４の実施の形態であるコンパイラが変換の対象とする呼び出し元プログラムであるアプリケーションプログラムモジュール１２１と、呼び出されるプログラムであるライブラリプログラムモジュール１５１と、その呼び出し処理時に行われる処理の一例を示す（図９（ｂ）、図９（ｃ））。

【０１４６】

この実施の形態では、呼び出し元アプリケーションプログラムモジュール単位、或いは呼び出し元アプリケーションプログラム単位、或いは呼び出し元アプリケーションプロセス単位、或いは呼び出し元アプリケーションタスク単位、或いは呼び出し元アプリケーション単位、或いは呼び出し元のプロセス単位に、或い

はまた呼び出し元のタスク単位に、ＣＣＴ（Ｃｏｎｔｅｘｔ　Ｃｏｎｔｒｏｌ　Ｔａｂｌｅ）と呼ばれる、各プログラムモジュールの固有のデータ領域へのアドレスを保存しているテーブルを備えている（図９（ａ））。

【０１４７】

呼び出し元のアプリケーションプログラムモジュール１２１では、アプリケーションプログラム１２１用ＣＣＴ（図９（ａ））を参照してアプリケーションプログラムモジュール１２１自体のデータ領域アドレスであるこの例では０ｘ６００を取り出し、この値をＤＰに設定し、この値をオフセットアドレスとして使用して外部変数アクセスすることによって、アプリケーションプログラムモジュール１２１用の固有データ領域１２５を使用する。

【０１４８】

このアプリケーションプログラムモジュール１２１からライブラリプログラムモジュール１５１に含まれる関数ｂａｒを呼び出すには、まず関数ｂａｒに引き渡す引数Ａと引数Ｂをスタックに保存する。

【０１４９】

次に、アプリケーションプログラムモジュール１２１ではこの関数ｂａｒから自分自身へ帰ってくるためのアドレス、即ち関数ｂａｒへのジャンプ命令の次のアドレスをスタックに保存し、関数ｂａｒに制御を移行する。

【０１５０】

制御の移行を受けた呼び出し先ライブラリプログラムモジュール１５１に含まれる関数ｂａｒでは、現行のＤＰの内容を保存する。現行のＤＰの内容とは呼び出し元のアプリケーションプログラムモジュール１２１用の固有データ領域アドレスの値であり、関数ｂａｒから呼び出し元のプログラムに戻った時、再度この値をＤＰに設定してアプリケーションプログラムモジュール１２１用の固有データ領域を使用するために保存しておくものである。

【０１５１】

次にこのＤＰの内容を参照してアプリケーションプログラムモジュール１２１用ＣＣＴへのポインタアドレスを取り出し、更にそのポインタアドレスを参照してアプリケーションプログラムモジュール１２１用ＣＣＴを取り出し、更にその

ＣＣＴの内容を参照して、関数ｂａｒでアプリケーションプログラムモジュール１２１が固有に使用するアプリケーションプログラムモジュール１２１用データ領域１５５のアドレス（この例では、０ｘ８００）を取り出し、その内容をＤＰレジスタに設定する（図９（ｃ））。

【０１５２】

これによって、呼び出されたライブラリプログラムモジュール１５１に含まれる関数ｂａｒでは、外部変数に対してアクセスする時、ＤＰレジスタの値をオフセットアドレスとして使用してアクセスするので、関数ｂａｒが使用する全ての外部変数は０ｘ８００がオフセットアドレスとして使用されてアクセスされ、アプリケーションプログラムモジュール１２１から呼び出される限りアプリケーションプログラムモジュール１２１用データ領域１５５が割り当てられる（図９（ｃ））。

【０１５３】

これによって、複数のアプリケーションプログラムモジュールから関数ｂａｒに対して再入が発生したとしても、外部変数で衝突が発生することは起こらない。

【０１５４】

関数ｂａｒから呼び出し元のアプリケーションプログラムモジュール１２１に戻る時には、ＤＰの内容を保存しておいた呼び出し元のアプリケーションプログラムモジュール１２１用の固有データ領域アドレスの値に戻してから、リタンする（図９（ｃ））。

【０１５５】

これによって、呼び出し元のアプリケーションプログラムモジュール１２１では再度、呼び出し元のアプリケーションプログラムモジュール１２１自体のデータ領域を使用することができる（図９（ｂ））。

【０１５６】

この実施の形態では、ライブラリプログラムモジュール１５１の呼び出し側でＤＰの設定を行わないため、呼び出し側のプログラムの負担を軽くすることができ、呼び出し回数が多い場合等に特に有効である。

【０１５７】

また図示していないが、図９（ｃ）で関数ｂａｒが「ｐｒｉｖａｔｅ」宣言が行われていれば、上記のようなＤＰの保存やアプリケーションプログラムモジュール１２１からの呼び出しで使用するｂａｒ用のＤＰを、ＣＣＴから取り出して設定等の処理を行わないことは上記第２の実施の形態と同じであり、「ｐｕｂｌｉｃ」宣言が行われているならば明示的に、ＤＰの保存やアプリケーションプログラムモジュール１２１からの呼び出しで使用するｂａｒ用のＤＰをＣＣＴから取り出して設定等の処理等が行われることを示す。これらは前記第２の実施の形態と同じであるから説明は省略する。

【０１５８】

更にまた図示していないが、このＣＣＴには前記ＤＰだけでなく、前記第３の実施の形態で説明した新しいＳＤＰ（０ｘ０Ｆ００）を、図２（ａ）のデータ領域１２５に格納して呼び出しプログラムモジュール１２１のプログラムコード領域１２３でその保存や設定を行うのではなく、例えば図９（ａ）のＣＣＴと同様のテーブルを用意して、そのテーブルに各プログラムモジュールの共有データ領域の先頭アドレスを格納しておき、呼び出されるライブラリプログラムモジュールである、例えば図９（ｃ）のライブラリプログラムモジュール１５１のプログラムコード領域１５３のステップの中で、そのＣＣＴと同様のテーブルとＣＣＴと同様のテーブルに格納されたＳＤＰを読み出して設定することもできる。

【０１５９】

これについても前記ＤＰの保存と設定と同じであるから詳細な説明は省略する。

【０１６０】

また、ＣＣＴと同様のテーブルは、ＣＣＴと異なり、実行環境で１つだけ保持するとしてもよい。

【０１６１】

また以上説明したような原始プログラム、目的プログラム、その他のプログラムの全てはコンピュータ読み取り可能な記録媒体に格納されて、コンピュータから読み出されて利用され、実行される、ネットワークによって通信される。

【０１６２】

同時にこのような翻訳を行うコンパイラもまたコンピュータ読み取り可能な記録媒体に格納されて、コンピュータから読み出されてコンパイル（原始プログラムから目的プログラムへの翻訳・変換）が実行され、利用され、ネットワークによって通信される。

【０１６３】

このコンパイラによってコンピュータが行うのはこのコンパイラによって規定されたコンパイル方法であり、このコンピュータハードウエアとコンパイラとは全体としてコンパイラ装置を構成する。

【０１６４】

【発明の効果】

以上説明したように、本発明によると、例えば、いわゆる機器組み込みプログラムのようにオブジェクトプログラムを格納することのできるメモリ領域が限られており、しかも近年の携帯情報機器のようにその限られたメモリ領域に高度且つ高機能な情報処理のためのオブジェクトプログラムを格納しなければならないような目的に適う、特にこのような目的に適うオブジェクトプログラムとして再入可能なオブジェクトプログラムを自動生成するためのコンパイラや、当該コンパイラを記録したコンピュータ読み取り可能な記録媒体や、そのための方法や装置を提供することができ、その効果は非常に大きい。

【図面の簡単な説明】

【図１】

本発明の第１の実施の形態であるコンパイラが変換するソースプログラムとそのコンパイラによって変換されたオブジェクトプログラムの構造を示す図

【図２】

本発明の第１の実施の形態のアプリケーションプログラムモジュール１２１がライブラリプログラムモジュール１５１を呼び出す時の処理手順を示す図

【図３】

本発明の第１の実施の形態のアプリケーションプログラムモジュール１２１がライブラリプログラムモジュール１５１を呼び出す時の処理手順を示す図

【図４】

本発明の第１の実施の形態のスタックの状態を示す図

【図５】

本発明の第３の実施の形態で呼び出されるプログラムモジュールにおける外部変数の宣言に関する記述を示す図

【図６】

本発明の第３の実施の形態で関数を呼び出す複数のプログラムモジュールが共通にアクセスする領域に関する記述を示す図

【図７】

本発明の第２の実施の形態で呼び出されるプログラムモジュールにおける関数の宣言に関する記述を示す図

【図８】

本発明の第２の実施の形態でライブラリプログラムモジュールが処理の一部を行わずに省略するための記述を示す図

【図９】

本発明の第４の実施の形態であるコンパイラが変換の対象とする呼び出し元プログラムと呼び出されるプログラムとその呼び出し時に行われる処理例を示す図

【符号の説明】

１２１　アプリケーションプログラムモジュール

１２３　プログラムコード領域

１２５　データ領域

１３１　アプリケーションプログラムモジュール

１３３　プログラムコード領域

１３５　データ領域

１５１　ライブラリプログラムモジュール

１５３　プログラムコード領域

１５５　アプリケーションプログラムモジュール１２１用データ領域

１５７　アプリケーションプログラムモジュール１３１用データ領域

５５１　ライブラリプログラムモジュール

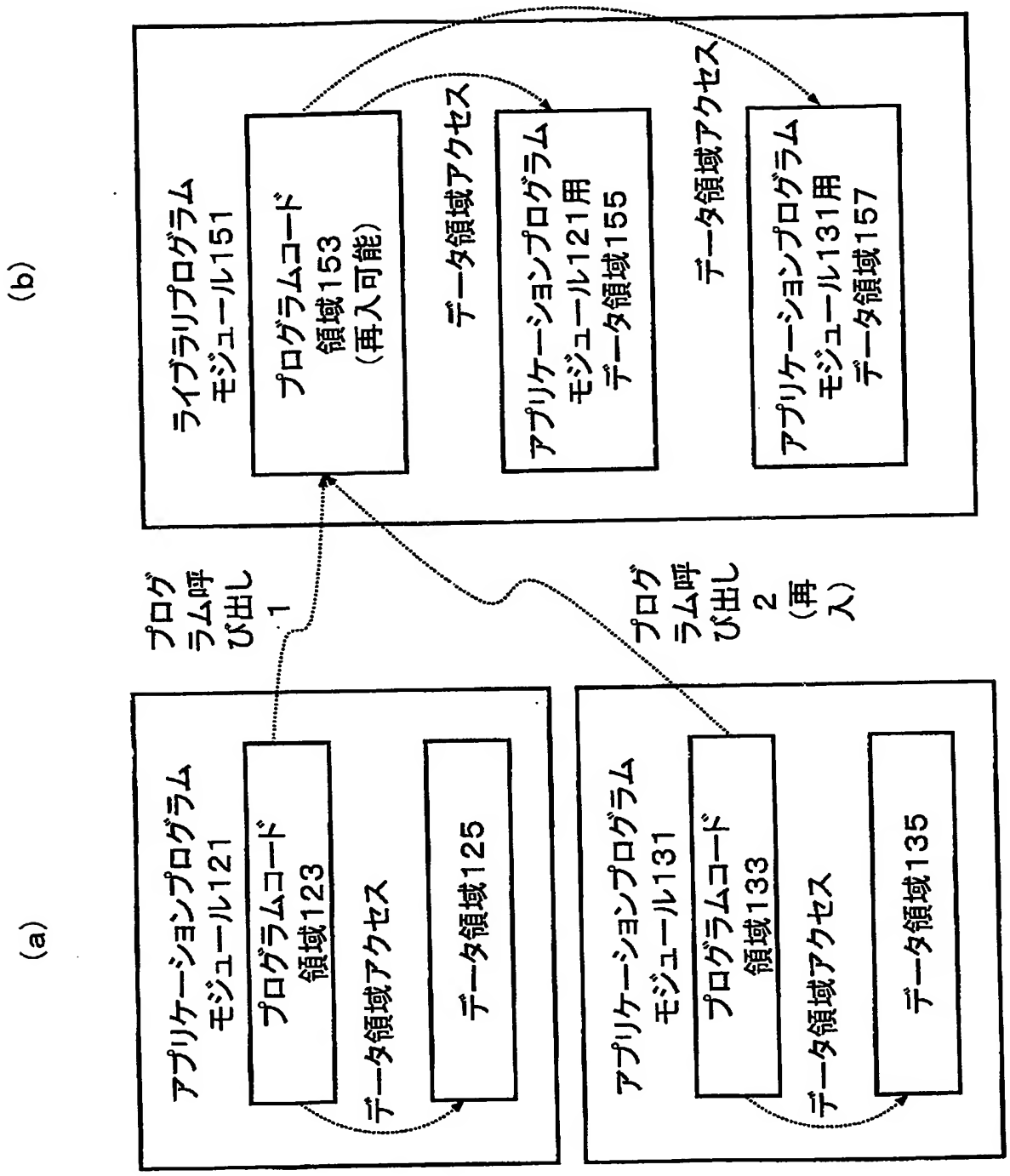
５５３　プログラムコード領域

５５５　アプリケーションプログラムモジュール１２１用データ領域

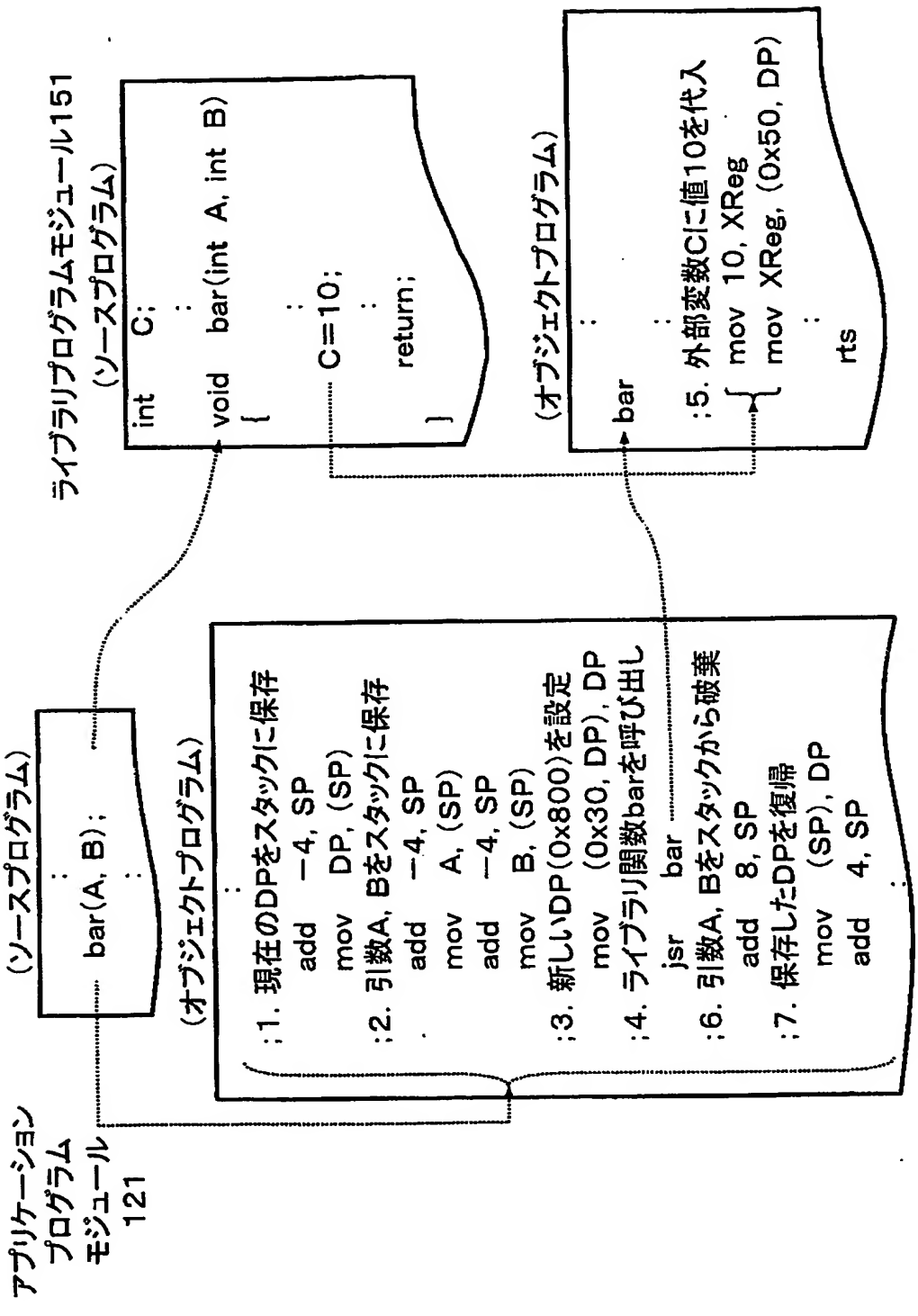
５５９　共用データ領域

【書類名】　図面

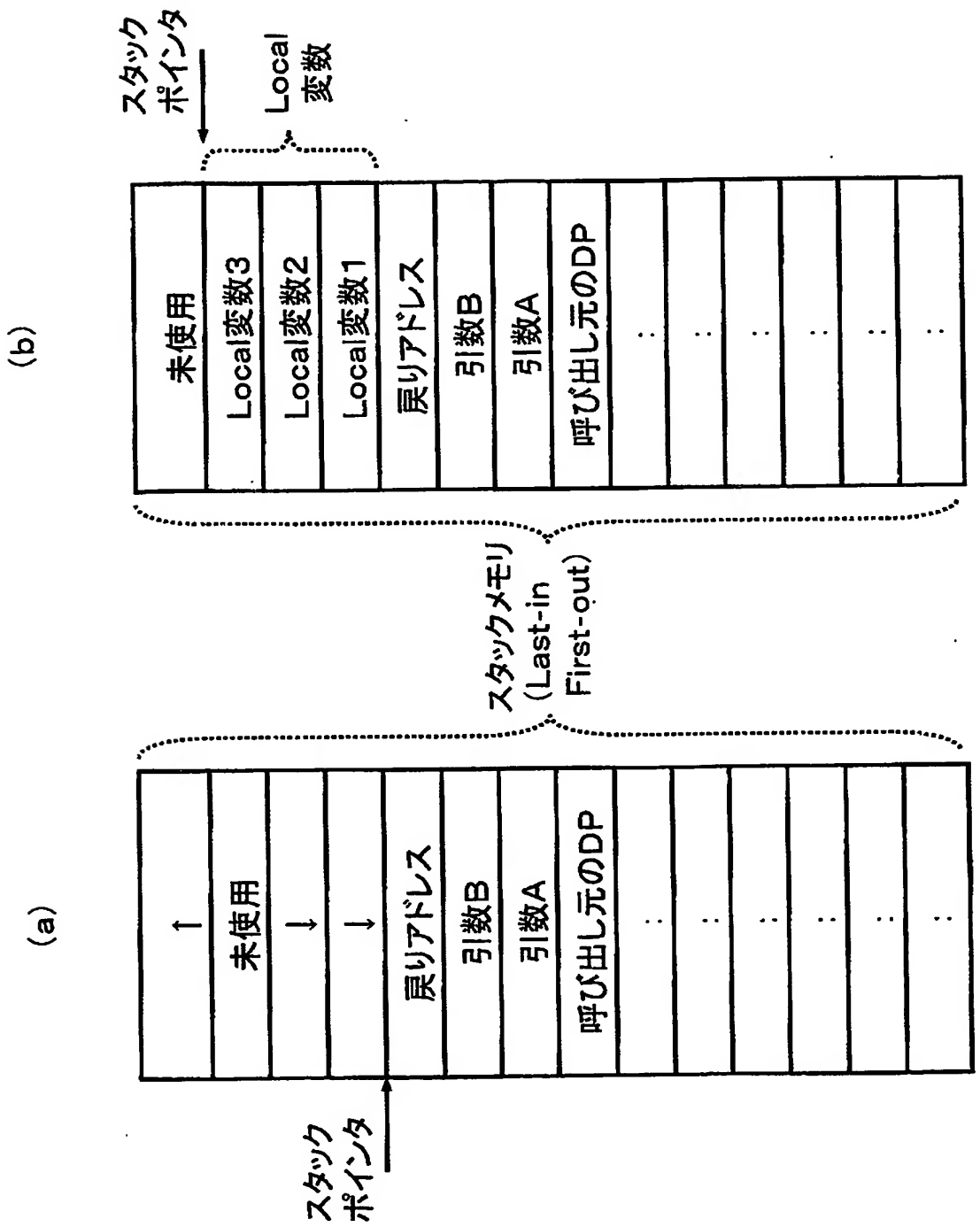
【図１】


(a)
アプリケーションプログラム
モジュール121
プログラムコード
領域123
データ領域アクセス
データ領域125
アプリケーションプログラム
モジュール131
プログラムコード
領域133
データ領域アクセス
データ領域135
プログラム呼び出し 1
プログラム呼び出し 2（再入）
(b)
ライブラリプログラム
モジュール151
プログラムコード
領域153
（再入可能）
データ領域アクセス
アプリケーションプログラム
モジュール121用
データ領域155
データ領域アクセス
アプリケーションプログラム
モジュール131用
データ領域157

【図３】


(a)
アプリケーション
プログラム
モジュール
121
(ソースプログラム)
bar(A, B);
(オブジェクトプログラム)
;1. 現在のDPをスタックに保存
add −4, SP
mov DP, (SP)
;2. 引数A, Bをスタックに保存
add −4, SP
mov A, (SP)
add −4, SP
mov B, (SP)
;3. 新しいDP(0x800)を設定
mov (0x30, DP), DP
;4. ライブラリ関数barを呼び出し
jsr bar
;6. 引数A, Bをスタックから破棄
add 8, SP
;7. 保存したDPを復帰
mov (SP), DP
add 4, SP
(b)
ライブラリプログラムモジュール151
(ソースプログラム)
int C;
void bar(int A, int B)
{
C=10;
return;
}
(オブジェクトプログラム)
bar
;5. 外部変数Cに値10を代入
mov 10, XReg
mov XReg, (0x50, DP)
rts

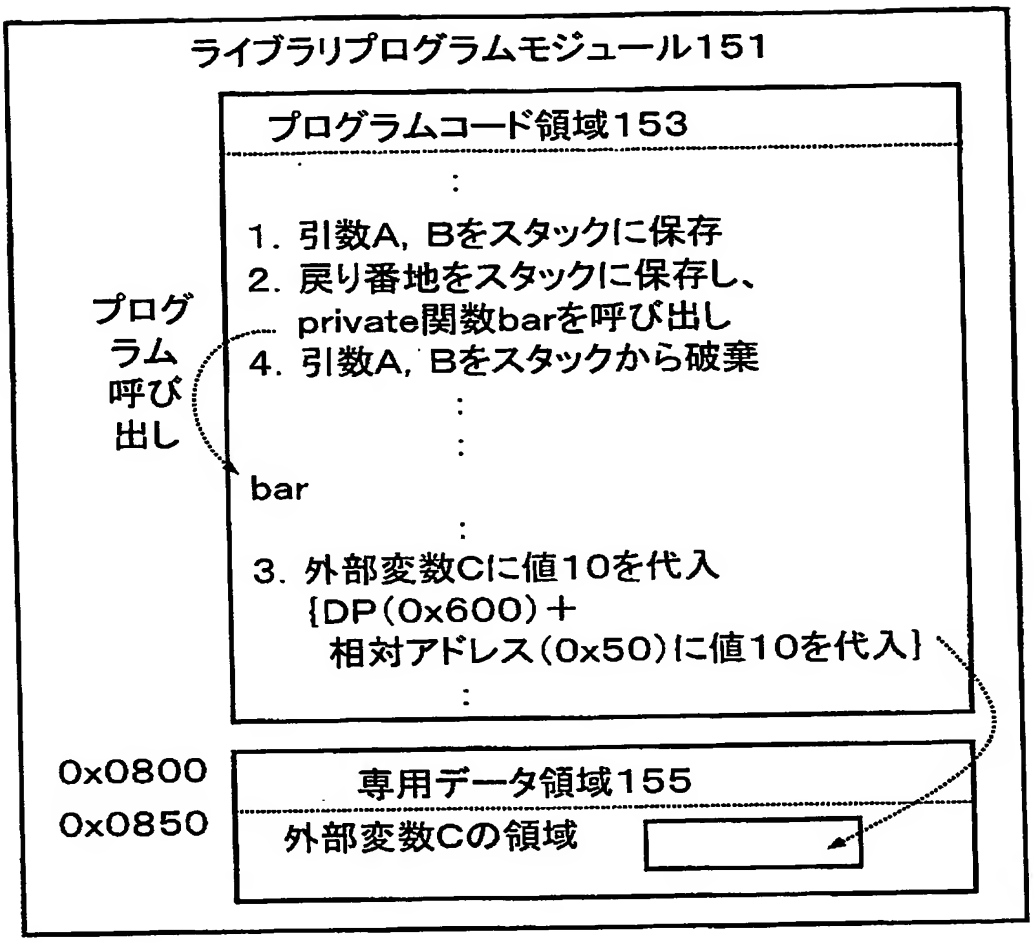
【図４】


(a)
(b)
スタックポインタ
Local変数
スタックメモリ
(Last-in First-out)
↑
未使用
↓
↓
戻りアドレス
引数B
引数A
呼び出し元のDP
未使用
Local変数3
Local変数2
Local変数1
戻りアドレス
引数B
引数A
呼び出し元のDP
スタックポインタ

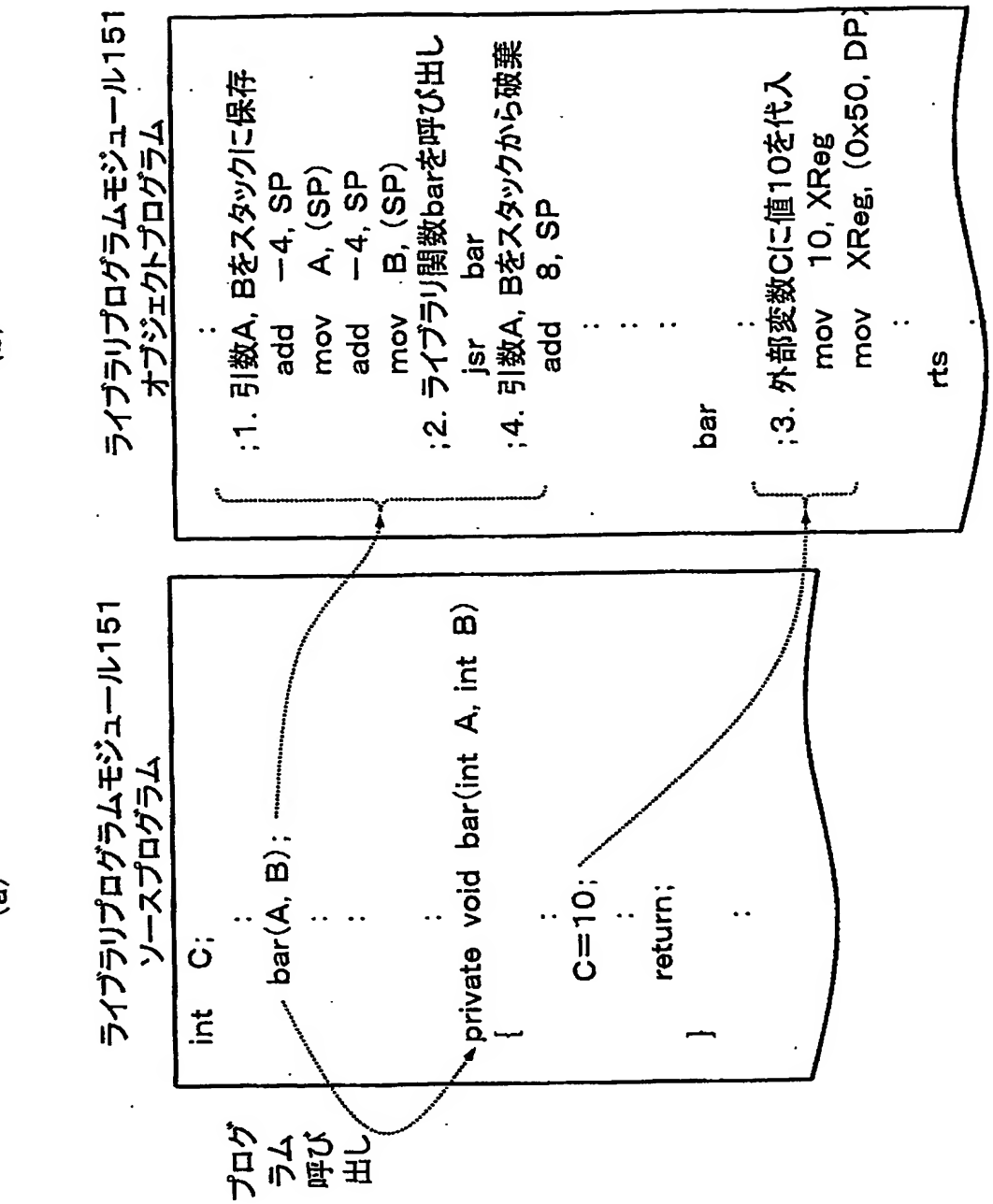
【図６】

(a)
ライブラリプログラムモジュール551
(ソースプログラム)
int C;
common int D;
void bar(int A, int B)
{
/* 外部変数Cに値10を代入 */
C=10;
/* 外部変数Dに値20を代入 */
D=20;
return;
}
(b)
ライブラリプログラムモジュール551
(オブジェクトプログラム)
bar
;外部変数Cに値10を代入
mov 10, XReg
mov XReg, (0x50, DP)
;外部変数Dに値20を代入
mov 20, XReg
mov XReg, (0x60, SDP)
rts

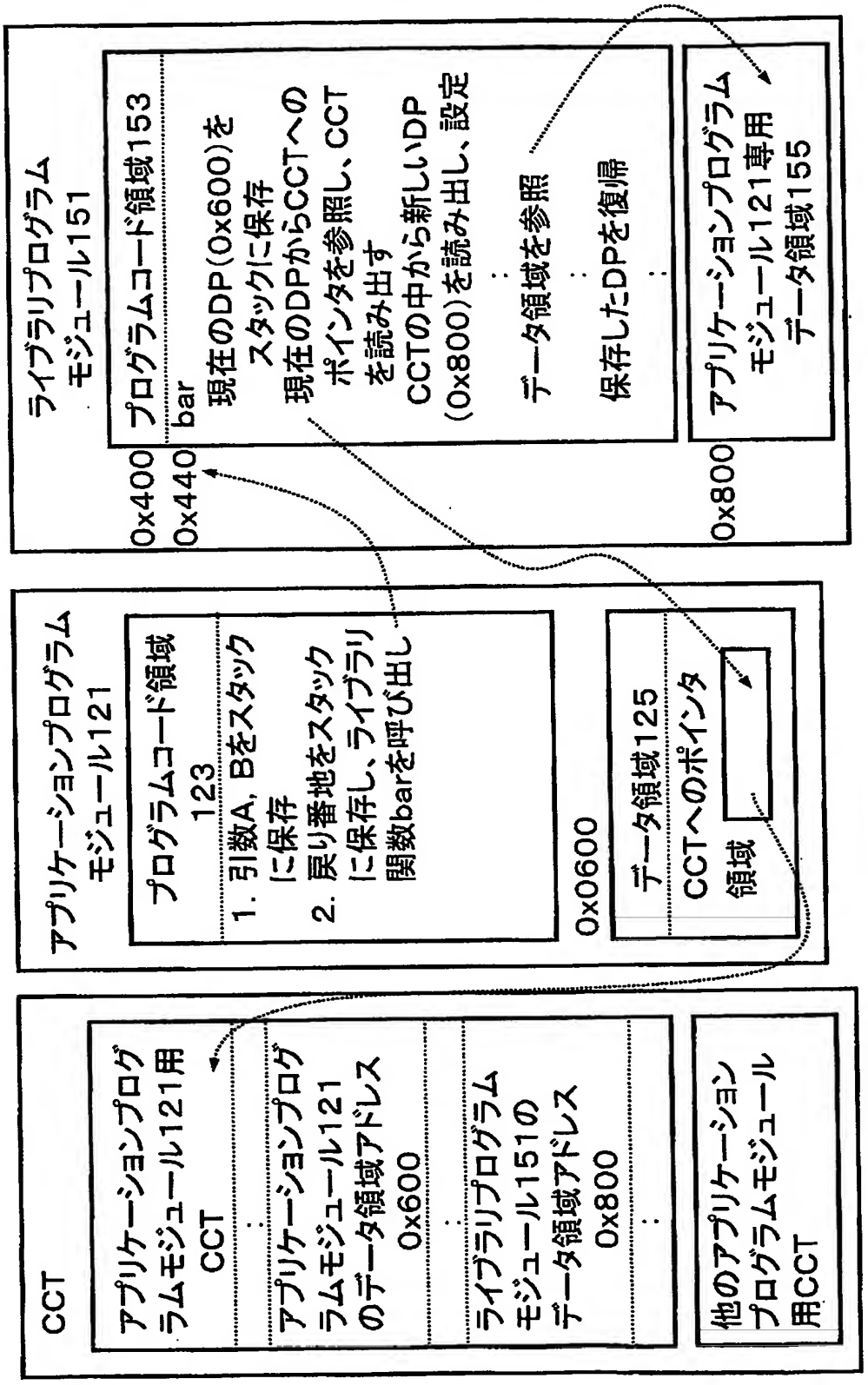
【図７】


ライブラリプログラムモジュール151
プログラムコード領域153
1. 引数A, Bをスタックに保存
2. 戻り番地をスタックに保存し、
private関数barを呼び出し
4. 引数A, Bをスタックから破棄
プログラム呼び出し
bar
3. 外部変数Cに値10を代入
{DP(0x600)＋
相対アドレス(0x50)に値10を代入}
0x0800
0x0850
専用データ領域155
外部変数Cの領域

【図８】


(a)
ライブラリプログラムモジュール151
ソースプログラム
int C;
bar(A, B);
private void bar(int A, int B)
{
C=10;
return;
}
プログラム呼び出し
(b)
ライブラリプログラムモジュール151
オブジェクトプログラム
:1. 引数A, Bをスタックに保存
add -4, SP
mov A, (SP)
add -4, SP
mov B, (SP)
:2. ライブラリ関数barを呼び出し
jsr bar
:4. 引数A, Bをスタックから破棄
add 8, SP
bar
:3. 外部変数Cに値10を代入
mov 10, XReg
mov XReg, (0x50, DP)
rts

【図９】


(a)
CCT
アプリケーションプログラムモジュール121用CCT
アプリケーションプログラムモジュール121のデータ領域アドレス
0x600
ライブラリプログラムモジュール151のデータ領域アドレス
0x800
他のアプリケーションプログラムモジュール用CCT
(b)
アプリケーションプログラムモジュール121
プログラムコード領域123
1. 引数A, Bをスタックに保存
2. 戻り番地をスタックに保存し、ライブラリ関数barを呼び出し
0x0600
データ領域125
CCTへのポインタ領域
(c)
ライブラリプログラムモジュール151
0x400
0x440
プログラムコード領域153
bar
現在のDP(0x600)をスタックに保存
現在のDPからCCTへのポインタを参照し、CCTを読み出す
CCTの中から新しいDP(0x800)を読み出し、設定
データ領域を参照
保存したDPを復帰
0x800
アプリケーションプログラムモジュール121専用データ領域155

【書類名】　要約書

【要約】

【課題】　再入可能なプログラムの作成は、その本質的部分を人間のプログラマの作業によって記述されるため、多くの時間を要しながら不完全な排他的制御しか実現することができず、その結果、プログラムを実行させた時に多くの誤動作を引き起こす原因となっていた。

【解決手段】　呼び出し元プログラムモジュールから他プログラムモジュールを呼び出すステップを前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップを含む目的プログラムのステップに変換し、呼び出される他プログラムモジュールに含まれるステップを前記設定された呼び出される他プログラム用データメモリ領域アドレスを参照してデータメモリ領域アクセスするステップを含む目的プログラムのステップに変換する。

【選択図】　図１

特願２００３－０００２７５

出願人履歴情報

識別番号　［０００００５８２１］

１．変更年月日　１９９０年　８月２８日
［変更理由］　新規登録
住　所　大阪府門真市大字門真１００６番地
氏　名　松下電器産業株式会社